Dominik Wawszczak
student id number: 440014
group number: 1

Parameterized Algorithms
assignment 1

Task 1

Let

$$V(G) \;=\; \{v_1, \ldots, v_n\} \quad \text{and} \quad S_n \;=\; \left\{\sigma \;:\; \sigma \in \{1, \ldots, n\}^{\{1,\ldots,n\}} \;\wedge\; \sigma \text{ is a bijection}\right\}.$$

Define

$$\text{cutwidth}_\sigma(i) \;=\; |\{(u, v) \;:\; u \in \{v_{\sigma_1} \ldots v_{\sigma_i}\} \;\wedge\; v \in \{v_{\sigma_{i+1}}, \ldots, v_{\sigma_n}\} \;\wedge\; (u, v) \in E(G)\}|,$$

where $\sigma \in S_n$ is any permutation. The objective is to find a permutation $\sigma \in S_n$ that minimizes the value

$$\max_{i \in \{1,\ldots,n-1\}} \text{cutwidth}_\sigma(i)$$

and to calculate this minimum.

Define the function

$$\text{out}(X) \;=\; |\{(u, v) \;:\; u \in X \;\wedge\; v \in V(G) \setminus X \;\wedge\; (u, v) \in E(G)\}|,$$

where X is any subset of $V(G)$. Then

$$\text{cutwidth}_\sigma(i) \;=\; \text{out}(\{v_{\sigma_1}, \ldots, v_{\sigma_i}\}).$$

For any given X, $\text{out}(X)$ can be calculated in time $n^{\mathcal{O}(1)}$.

We will use dynamic programming over subsets. Define

$$\text{dp}(X) \;=\; \min\left\{\max_{i \in \{1,\ldots,|X|-1\}} \text{cutwidth}_\sigma(i) \;:\; \sigma \in S_n \;\wedge\; \{v_{\sigma_1}, \ldots, v_{\sigma_{|X|}}\} = X\right\},$$

where X is any subset of $V(G)$. Then

$$\begin{aligned}
\text{dp}(\emptyset) \;&=\; 0, \\
\text{dp}(X) \;&=\; \min\{\max(\text{dp}(X \setminus \{x\}), \text{out}(X \setminus \{x\})) \;:\; x \in X\}.
\end{aligned}$$

The answer to the problem is $\text{dp}(\{1, \ldots, n\})$. To compute dp for all subsets of $V(G)$, we iterate over subsets in non-decreasing order of size. The time complexity of this algorithm is $2^n \cdot n^{\mathcal{O}(1)}$.

Task 2

Lemma 1 A set of points S, in which no three points are collinear, does not form the vertices of a convex polygon if and only if there exist points $A, B, C, D \in S$ such that D lies inside $\triangle ABC$.

Proof of lemma 1 The implication „to the left” is obvious, so we only need to prove the implication „to the right”. Let $\{H_1, \ldots, H_h\}$ be the convex hull of the set S, with the assumption that these points are ordered counterclockwise along the hull. Let D be any point in S that does not belong to the hull, and let $A = H_1$. There exists exactly one $i \in \{2, \ldots, h-1\}$ such that points $H_2, \ldots, H_i$ lie on one side of the line AD, while points $H_{i+1}, \ldots, H_h$ lie on the other side. Taking $B = H_i$ and $C = H_{i+1}$, we will obtain the desired points.

According to lemma 1, if there exist points $A, B, C, D \in S$ such that D lies inside $\triangle ABC$, then at least of these points must be removed from S. This observation leads to the following algorithm:

Algorithm 1 ConvexDeletion

procedure ConvexDeletion(S, k)
  if no four points $A, B, C, D \in S$ satisfy that D lies inside $\triangle ABC$ **then**
    return true
  end if
  if $k \leqslant 0$ **then**
    return false
  end if
  Choose points $(A, B, C, D) \in S$ such that D lies inside $\triangle ABC$
  return ConvexDeletion($S \setminus \{A\}, k-1$) **or** ConvexDeletion($S \setminus \{B\}, k-1$) **or** ConvexDeletion($S \setminus \{C\}, k-1$) **or** ConvexDeletion($S \setminus \{D\}, k-1$)
end procedure

Finding such quadruples of points A, B, C, D can easily be done in $\mathcal{O}(n^4)$ time by examining all quadruples of points, calculating the relevant cross products for each, and comparing their signs. This can also be done in $\mathcal{O}(n \log n)$ time by computing the convex hull using Graham's algorithm and applying the constructive proof of lemma 1.

The depth of the recursion tree fro of the algorithm 1 is at most k, since with each recursive call, the parameter k decreases by 1. Each node of this tree has at most four children, which gives us an upper bound on the number of nodes in the tree:

$$\sum_{i=0}^{k} 4^i = \frac{4^{k+1} - 1}{3} = \mathcal{O}(4^k).$$

Therefore, the overall complexity of the algorithm 1 is $\mathcal{O}(4^k) \cdot n^{\mathcal{O}(1)}$.